John Dumas
MR 3421 Cloud Physics
Final Project
14 June, 1999

# A Cloud Drop Model

# Physics: Introduction

This project is a simple model of a falling cloud drop. The drop grows by the collision and coalescence process within the boundaries of the cloud. Additionally, the drop grows by condensation or evaporates throughout the model run based on the relative humidity. The reference for this model is the text *A Short Course in Cloud Physics (3rd ed)* by R.R. Rogers and M. K. Yau (hereafter R&Y). It is a "simplified" model because there were many assumptions or limitations imposed on the model by either physics or my programming ability. These limitations, as well as the rest of the cloud physics are described below. A summary of simplifications is provided as Appendix 1.

# Collisions

The primary mechanism for drop growth is collision and coalescence. Simply put - large drops fall faster than smaller drops and they acquire a portion of the smaller drops along their path. This process is expressed mathematically as R&Y equation 8.15 (p131)

$$\frac{dR}{dt} = \frac{\bar{E}M}{4\tilde{n}_{L}} u(R)$$

where

R is the radius of the large cloud drop,

u(R) is the terminal fall speed of the large drop,

E is the collision efficiency of the large drop falling relative to smaller droplets, and

$\rho_L$ is the density of the liquid.

I chose the drop to be pure water so the density would remain a constant 1000 kg/m$^3$. I suppose this could be a variable quantity, but I thought that would add unnecessary complexity to the program. I used R&Y Table 8.2 to determine the collision efficiency.

TABLE 8.2. *Collision Efficiency* E *for Drops of Radius* R *Colliding with Droplets of Radius* r. *(Data for* R < *50 μm adapted from Klett and Davis, 1973; for 50 μm* ≤ R ≤ *500 μm from Beard and Ochs, 1984; for* R > *500 μm from Mason, 1971)*

| | r (μm) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| R (μm) | 2 | 3 | 4 | 6 | 8 | 10 | 15 | 20 | 25 |
| 10 | 0.017 | 0.027 | 0.037 | 0.052 | 0.052 | | | | |
| 20 | * | 0.016 | 0.027 | 0.060 | 0.12 | 0.17 | 0.17 | | |
| 30 | * | * | 0.020 | 0.13 | 0.28 | 0.37 | 0.54 | 0.55 | 0.47 |
| 40 | * | * | 0.020 | 0.23 | 0.40 | 0.55 | 0.70 | 0.75 | 0.75 |
| 50 | — | — | 0.030 | 0.30 | 0.40 | 0.58 | 0.73 | 0.75 | 0.79 |
| 60 | — | 0.010 | 0.13 | 0.38 | 0.57 | 0.68 | 0.80 | 0.86 | 0.91 |
| 80 | — | 0.085 | 0.23 | 0.52 | 0.68 | 0.76 | 0.86 | 0.92 | 0.95 |
| 100 | — | 0.14 | 0.32 | 0.60 | 0.73 | 0.81 | 0.90 | 0.94 | 0.96 |
| 150 | 0.025 | 0.25 | 0.43 | 0.66 | 0.78 | 0.83 | 0.92 | 0.95 | 0.96 |
| 200 | 0.039 | 0.30 | 0.46 | 0.69 | 0.81 | 0.87 | 0.93 | 0.95 | 0.96 |
| 300 | 0.095 | 0.33 | 0.51 | 0.72 | 0.82 | 0.87 | 0.93 | 0.96 | 0.97 |
| 400 | 0.098 | 0.36 | 0.51 | 0.73 | 0.83 | 0.88 | 0.93 | 0.96 | 0.97 |
| 500 | 0.10 | 0.36 | 0.52 | 0.74 | 0.83 | 0.88 | 0.93 | 0.96 | 0.97 |
| 600 | 0.17 | 0.40 | 0.54 | 0.72 | 0.83 | 0.88 | 0.94 | 0.98 | † |
| 1000 | 0.15 | 0.37 | 0.52 | 0.74 | 0.82 | 0.88 | 0.94 | 0.98 | † |
| 1400 | 0.11 | 0.34 | 0.49 | 0.71 | 0.83 | 0.88 | 0.94 | 0.95 | † |
| 1800 | 0.08 | 0.29 | 0.45 | 0.68 | 0.80 | 0.86 | 0.96 | 0.94 | † |
| 2400 | 0.04 | 0.22 | 0.39 | 0.62 | 0.75 | 0.83 | 0.92 | 0.96 | † |
| 3000 | 0.02 | 0.16 | 0.33 | 0.55 | 0.71 | 0.81 | 0.90 | 0.94 | † |

—Collision efficiency less than 0.01.
*Value cannot be determined accurately from available data.
†Value close to one.

This is the first point where simplifications had to be made. Rather than allow for small cloud droplet particles of varying sized I fixed the ambient particle size at ten microns and duplicated the appropriate column of this table in the model through a series of "if R size" statements. Modifications to the program could include making the ambient droplets of a fixed, but selected size, or purely random.

Rather than restrict the cloud drop size to under 3mm (3000μm), I chose a uniform efficiency of 0.81 for any drop over 2.4mm relative to the 10μm surrounding particles.

The next area of simplification came in the calculation of fall speed, which proved to be problematic.

## Terminal Velocity

Individual liquid water droplets move with the velocity of the air surrounding them plus their fall velocity relative to the air. Maximum fall velocity or "terminal fall speed" will occur when the downward acting force of gravity is balanced by the retarding drag force acting on the droplet. From

Newton's second law (F=ma), when $F_g=F_R$, the resultant force is zero - so acceleration will be zero and the droplet will continue to fall at a constant velocity. This constant fall speed is a critical value in this model as it defines the drop's motion as well as factoring in the chance of a collision event.

A simplified formula for fall velocity that assumes a spherical drop of constant radius is R&Y equation 8.4 (p125).

$$u = \frac{2}{9}\frac{r^2 g\tilde{n}_L}{(C_D Re/24)\grave{i}}$$

where:

u is the terminal fall speed,

r is the radius of the large cloud drop,

g is the acceleration due to gravity ($9.8 m/s^2$),

$\rho_L$ is the density of the liquid (assumed >> density of air)

$C_D$ is the drag coefficient characterizing the flow,

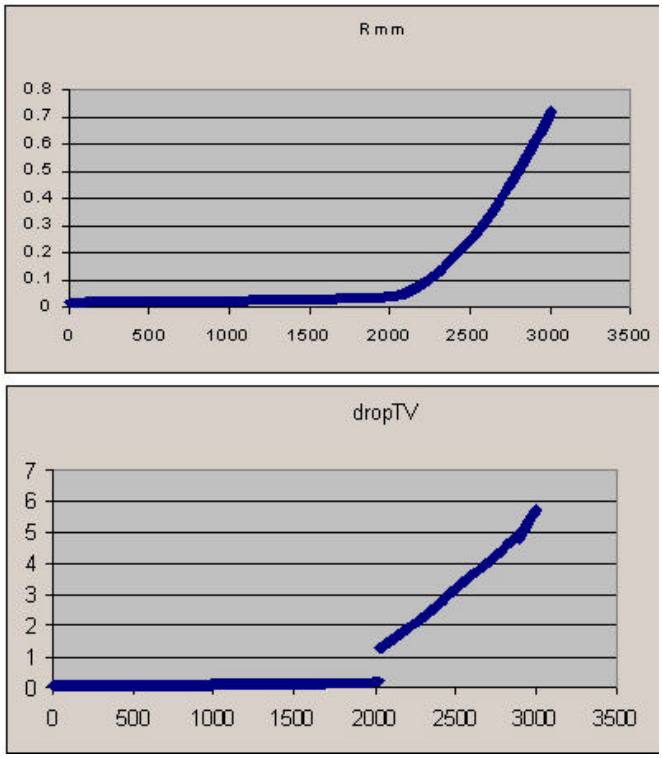
$\mu$ is the dynamic viscosity of the fluid, and

Re is the Reynolds number $Re=2\rho ur/\mu$.

Simplifications used to solve this equation varied with the size of the drop. For "very small" Reynolds numbers the Stokes solution to the flow field around a sphere $(C_D Re/24)=1$ and the speed is proportional to the square of the radius. For "sufficiently high" Reynolds numbers the drag becomes independent of Re and the speed tends with the square root of the radius. In addition to theoretical relations the text also included a table of observed fall speeds for radii of 0.1 to 5.8 mm. Appendix 2 includes the table, a plot of this table and the linearizations I used for r > 2mm. All of these relations are summarized in the table below:

| Droplet Radius | Terminal Speed | k | Notes |
|---|---|---|---|
| <30μm | $u=k_1r^2$ | $1.19x10^6$ $cm^{-1}$ $s^{-1}$ | "Stokes Solution"- for very small Re ($C_DRe/24$)=1 |
| 40μm<r<0.6 mm | $u=k_3r$ | $8x10^3$ $s^{-1}$ | |
| 0.6mm<r<2mm | $u=k_2r^{1/2}$ | $2.2x10^3(\rho o/\rho)^{1/2}$ $cm^{1/2}$ $s^{-1}$ | High Re, $C_D$ independent of Re and =0.45 (by experiment) |
| 0.6mm<r<2mm | $u=k_2r^{1/2}$ | $2.01x10^3$ $cm^{1/2}$ $s^{-1}$ | Based on tabulated data |
| r>2mm | NA | NA | Based on my linear approximations to R&Y table 8.1 |

The problem with trying to use each of these equations over a range drops of widely varying size is that they are not continuous. This was most apparent when moving from the first to the second equation around 40 microns as shown in the plot below:




The top graph is a plot of radius (y-axis) growing as a function of time (x-axis) from one of the early model runs. The bottom graph is the terminal fall speed (y-axis) for this same time. Note the jump in speed at approximately 2000 seconds as the radius exceeded >40 microns and the second equation began to be used.

There are other jumps, but none so detrimental to model performance.

I used a weighted average of solutions to each of the equations to resolve this, but the result is that my fall speeds for small drops are significantly higher than those modeled in the text. I would have preferred to use the data from table 8.1 throughout the model, but it only went down to drops 100 microns in radius. This is the main detraction of the model as I see it.

## Condensation

The equation for drop growth by condensation (or drop shrinkage by evaporation) is R&Y's innocent looking equation 7.19 (p103)

$$r(t) = \sqrt{r_o^2 + 2\text{å}\ t}$$

where r is the drop radius, r(t) indicates the radius as a function of time (t) and $r_o$ is the initial drop radius. Things begin to get difficult when calculating ε, which is a condensation growth parameter obtained from

$$\text{å} = (S - 1)\text{å}_1$$

S here is the ambient saturation ratio, which is the ratio of vapor pressure at the drop compared to the saturation vapor pressure at some distance:

$$S = \frac{e}{e_s(T)_\infty}$$

$e_s(T)$ for water can be calculated within 1% over the temperature range $-30^oC \leq T \leq +35^oC$ by the empirical formula (R&Y eqn 2.17, p.16)

$$e_s(T) = 6.112\exp\left(\frac{17.67T}{T + 243.5}\right)$$

and e can be obtained from the equation of state (R&Y eqn 2.1, p.12)

$$e = \tilde{n}_v R_v T$$

where $\rho_v$ is the vapor density, $R_v$ is the individual gas constant for water vapor (461.5 J $kg^{-1}$ $K^{-1}$) and T is the absolute temperature. Rather than prompting for input related to vapor pressure, typical values of which I'm a bit sketchy on, I opted for a direct entry for the saturation ratio value S. An S value of 1.01 indicating 1% supersaturation relays much more physical information to me than anything you could specify about a vapor pressure value.

The second part of calculating ε involved the normalized condensation growth parameter $\varepsilon_1$ which is a function of a thermodynamic term ($F_k$) associated with heat conduction and a second term ($F_d$) associated with vapor diffusion.

$$\varepsilon_1 = \frac{1}{F_k + F_d}, \text{ with } F_k = \left( \frac{L}{R_v T} - 1 \right) \frac{L \rho_L}{KT} \text{ and } F_d = \frac{\rho_L R_v T}{D e_s(T)}$$

where:

L is the latent heat of condensation, $\rho_L$ is the density of the liquid, T is the absolute temperature, $R_v$ is the individual gas constant for water vapor (461.5 J $kg^{-1}$ $K^{-1}$), $e_s(T)$ is described by the empirical formula listed above as (R&Y eqn 2.17, p.16), K is the coefficient of thermal conductivity of air, and D is the coefficient of diffusion of water vapor in air.

Values for L, D, and K were obtained from R&Y tables:

TABLE 2.1. *Saturation Vapor Pressures Over Water and Ice, and Latent Heats of Condensation and Sublimation*

| T(°C) | $e_s$(Pa) | $e_i$(Pa) | L (J/g) | $L_s$(J/g) |
|---|---|---|---|---|
| −40 | 19.05 | 12.85 | 2603 | 2839 |
| −35 | 31.54 | 22.36 | | |
| −30 | 51.06 | 38.02 | 2575 | 2839 |
| −25 | 80.90 | 63.30 | | |
| −20 | 125.63 | 103.28 | 2549 | 2838 |
| −15 | 191.44 | 165.32 | | |
| −10 | 286.57 | 259.92 | 2525 | 2837 |
| −5 | 421.84 | 401.78 | | |
| 0 | 611.21 | 611.15 | 2501 | 2834 |
| 5 | 872.47 | | 2489 | |
| 10 | 1227.94 | | 2477 | |
| 15 | 1705.32 | | 2466 | |
| 20 | 2338.54 | | 2453 | |
| 25 | 3168.74 | | 2442 | |
| 30 | 4245.20 | | 2430 | |
| 35 | 5626.45 | | 2418 | |
| 40 | 7381.27 | | 2406 | |

TABLE 7.1. *Values of Dynamic Viscosity μ and Coefficient of Thermal Conductivity K of Air, and Coefficient of Diffusion D of Water Vapor in Air. (From Houghton, 1985)*

| T(°C) | $\mu$ (kg $m^{-1}$ $s^{-1}$) | K (J $m^{-1}$ $s^{-1}$ $K^{-1}$) | D ($m^2$ $s^{-1}$) |
|---|---|---|---|
| −40 | $1.512 \times 10^{-5}$ | $2.07 \times 10^{-2}$ | $1.62 \times 10^{-5}$ |
| −30 | $1.564 \times 10^{-5}$ | $2.16 \times 10^{-2}$ | $1.76 \times 10^{-5}$ |
| −20 | $1.616 \times 10^{-5}$ | $2.24 \times 10^{-2}$ | $1.91 \times 10^{-5}$ |
| −10 | $1.667 \times 10^{-5}$ | $2.32 \times 10^{-2}$ | $2.06 \times 10^{-5}$ |
| 0 | $1.717 \times 10^{-5}$ | $2.40 \times 10^{-2}$ | $2.21 \times 10^{-5}$ |
| 10 | $1.766 \times 10^{-5}$ | $2.48 \times 10^{-2}$ | $2.36 \times 10^{-5}$ |
| 20 | $1.815 \times 10^{-5}$ | $2.55 \times 10^{-2}$ | $2.52 \times 10^{-5}$ |
| 30 | $1.862 \times 10^{-5}$ | $2.63 \times 10^{-2}$ | $2.69 \times 10^{-5}$ |

Note: The tabulated values of *D* are for a pressure of 100 kPa. Because *D* is proportional to $\mu/\rho$, it follows that *D* is inversely proportional to pressure for a given temperature. To obtain *D* for an arbitrary pressure *p* (kPa), the tabulated value must therefore be multiplied by (100/*p*).

These tables led to yet another simplification, as I had to choose a temperature to get values. Since the model was not going to include any ice processes, and since I didn't want to program these tables, I chose a constant value of $10^{o}$C for the entire model environment. Table 7.1 also is based on a constant pressure of 100kPa, so a conversion is required for any pressure other than 1000mb. I decided to leave that input to the user, but it is a simplification in that the entire model atmosphere is at this uniform pressure. I used 850mb as a default. With these fixed values, normalized condensation growth parameter is $1.085 \times 10^{-4}$ mm$^2$/s. At a typical supersaturation of only 1% it would then take thousands of seconds to get any appreciable growth by condensation, and this agrees with observational data. What began as the simple looking (7.19) was in fact the more complicated differential equation (7.17) for growth:

$$r\frac{dr}{dt} = \frac{S-1}{\left[\left(\frac{L}{R_v T} - 1\right)\frac{L\tilde{n}_L}{KT} + \frac{\tilde{n}_L R_v T}{De_s(T)}\right]}$$

# The Model

## Modes

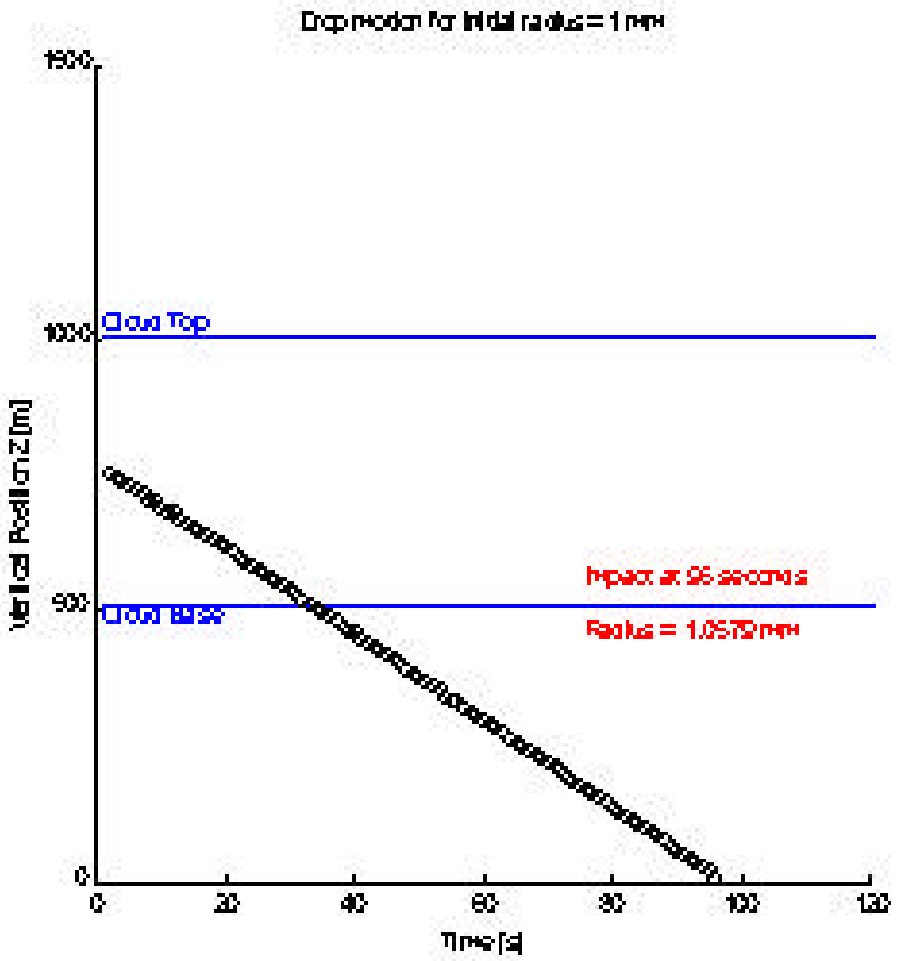
The model can be used to simulate a collision-only atmosphere, condensation-only atmosphere, or both. Since collisions only occur in the area between the cloud base and cloud top, if these are properly set a condensation only model results. For example, the cloud case could be set at 0 meters and cloud top at 1, with the drop initially at 500m. Since the axes of the plot are determined by the length of the run and the height of the cloud top it's better to make the cloud artificially too low than too high, to preserve the details. Since the condensation growth is determined by $\dot{a} = (S-1)a_1$, if the saturation ratio is set at 1 for all model levels, then this parameter is zero and a pure collision model results.

## Mechanics

The entire MATLAB code is provided as Appendix 3. Following a choice of default parameter input or user input, a few basic calculations are performed and the key variables are established. "LengthOfTime" is the value in seconds of the run. The model time step is one second. "Rmm" is an array holding the value of the radius of the drop (in millimeters) at the end of every second. "Zpos" is an array that holds the value of the drop's vertical position Z in meters at the end of every second.

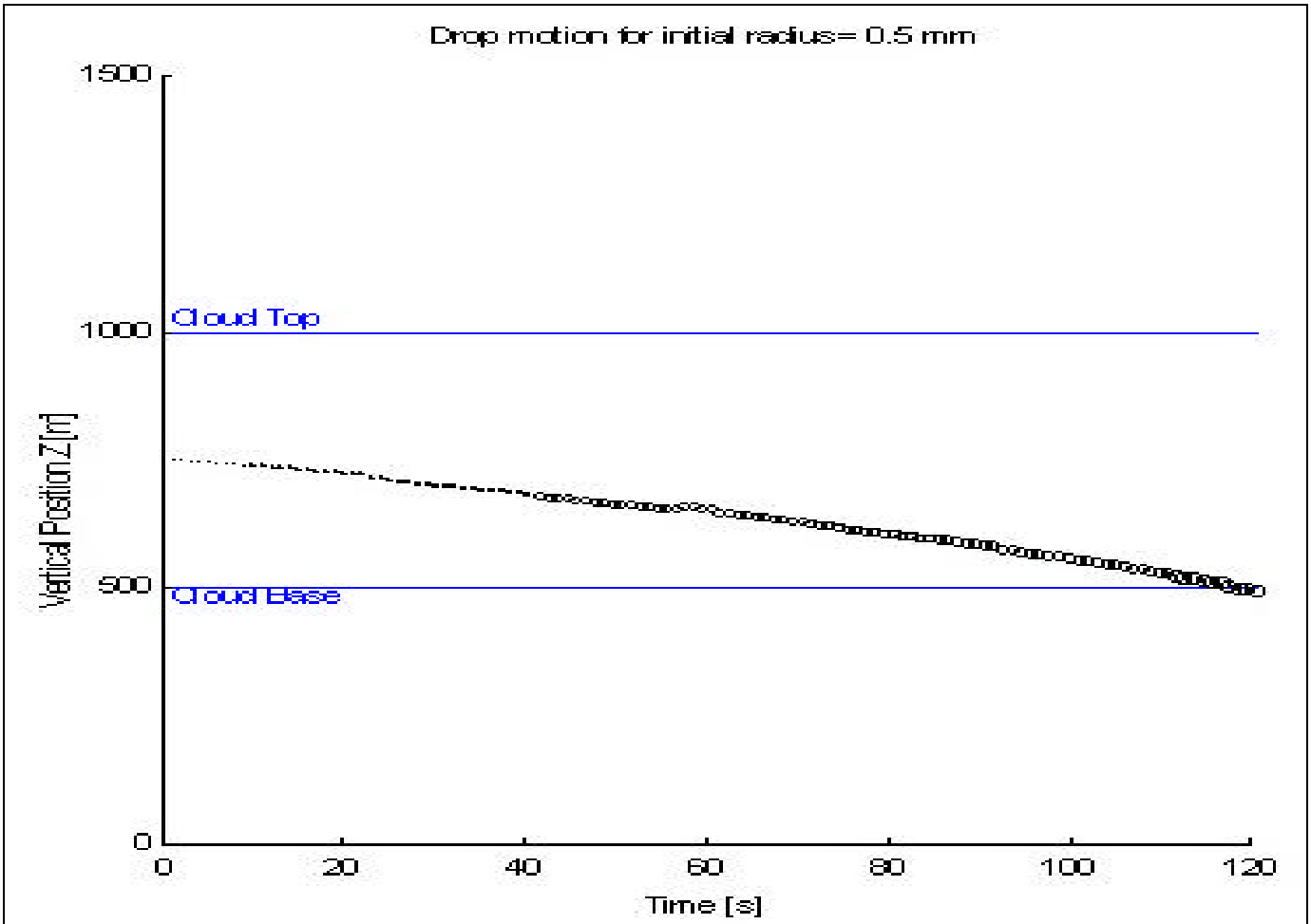
The program's calculations are performed in a loop beginning with the initial values. The first calculation is the terminal velocity for a drop of this radius. Note that this means in subsequent iterations, as the radius changes the particle's velocity is instantaneously changing to this new terminal velocity. Since the growth rates are quite small and the time step is only one second this isn't too bad of an assumption, but it remains as a simplification that could be refined in subsequent versions. The model then grows the drop by collisions if it is within the cloud, and then applies growth or evaporation based on saturation conditions for the altitude. The loop returns to the beginning and assigns the values of Rmm and zpos at the start of this next second what they were at the end of the previous second. The

loop continues until the LengthOfRun has been reached and then the program draws the plot of the motion. A plot based on the default conditions is below.




Unfortunately, although the figures would import directly into MS Word 97 from MATLAB, the file sizes were quite large and they wouldn't print correctly so I turned them into JPEG files in Adobe PhotoDeluxe and they got a bit blurry. This plot simply showed that the default drop of 1mm fell in a fairly linear pattern from its initial height of 750m to the ground in 96 seconds, growing by condensation and collision within the cloud, then evaporating to a final size of 1.0579 mm along the way.

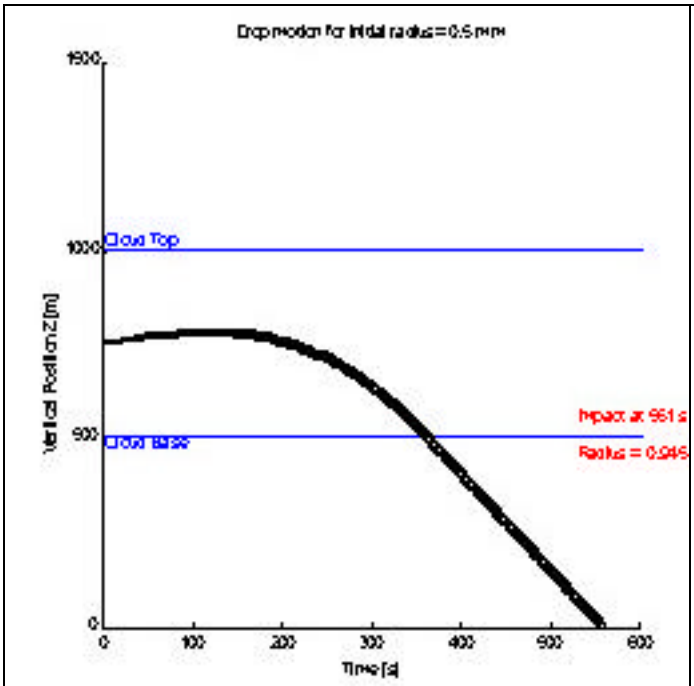
The drop is drawn as a text lowercase "o" placed at appropriate time and height. In order to convey the sense of growth I scaled the size of the letter from 2 to 20 point font from the drop at its smallest to the drop at its largest. A good example of a model run showing this growth is below:

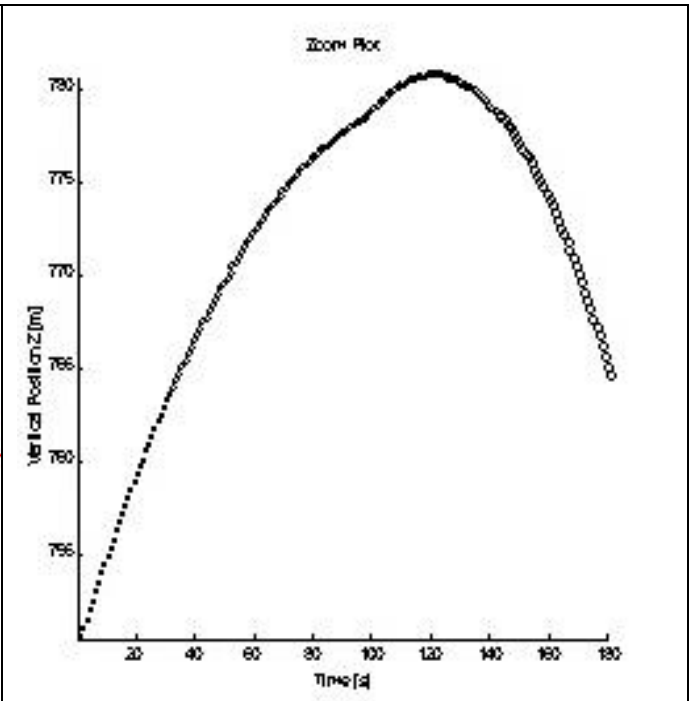



In Cloud Growth

```
The following information applies for this run:
The constant updraft velocity [m/s]              3.00
The cloud liquid water content M [kg/m3]         0.002
The cloud top height [m]                      1000.0
The cloud base height [m]                      500.0
The avg atmospheric pressure [mb]              850.0
The cloud saturation ratio S                     1.02
The rel above cloud humidity [percent]           0.50
The rel below cloud humidity [percent]           0.80
```

Obviously, if I wanted to see this kind of detail in a plot that was potentially thousands of seconds long I would have to make a zoom function. An example of this is shown below for a six-hundred second profile with a zoom-plot of it's initial growth and arc-like trajectory from 1 to 160 seconds.

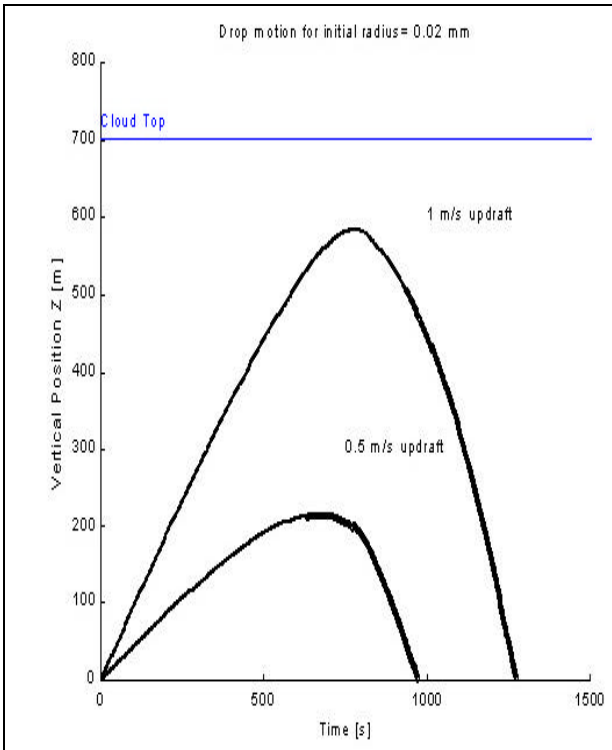



Falling Profile

```
The following information applies for this run:
The constant updraft velocity [m/s]              5.00
The cloud liquid water content M [kg/m3]         0.001
The cloud top height [m]                      1000.0
The cloud base height [m]                      500.0
The avg atmospheric pressure [mb]              850.0
The cloud saturation ratio S                     1.01
The rel above cloud humidity [percent]           0.50
The rel below cloud humidity [percent]           0.80
```

I wanted to "prove" the model by comparing it to one of the figures in R&Y. I chose to attempt to duplicate figure 8.5, that showed the change in altitude and time for a fixed size drop subject to an increasing updraft. The shape of my model output matches almost exactly, which I found quite heartening. The scale is off by about one-third, and I know this to be because of the terminal velocity problem for small drops as explained earlier, so I considered this plot proof enough for this version of the model. Suggestions for improvements to either the physics or code are welcome.




Overlaid plots of model output for two runs with an initial radius of 20μm, cloud liquid water content 1g/m$^3$, all cloud droplets of 10μm and coalescence efficiency from Rogers and Yau Table 8.2. Saturation was set at 100% so there would be no growth by condensation or expansion.

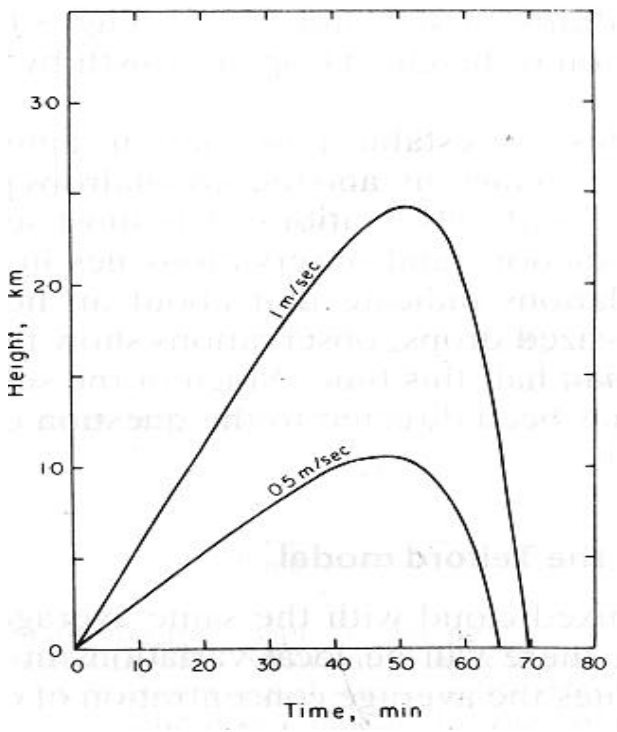



FIG. 8.5. Drop trajectories calculated for the collision efficiencies of Table 8.2 and Fig. 8.3, assuming a coalescence efficiency of unity. Initial drop radius 20 $\mu$m. Cloud water content 1 g/m$^3$; all cloud droplets of 10 $\mu$m radius.

(Rogers and Yau, p.133)

# Appendix 1: Summary of Simplifications/Problems

Cloud drops are pure water

Density of cloud drops >> density of air

Cloud drops are perfect spheres

All ambient cloud droplets are uniformly10μm

Terminal fall speed solutions are too high for small drops

Terminal speeds are reached instantaneously

Collision efficiency is estimated above a radius of 3mm

Temperature is fixed at $10^{o}$C

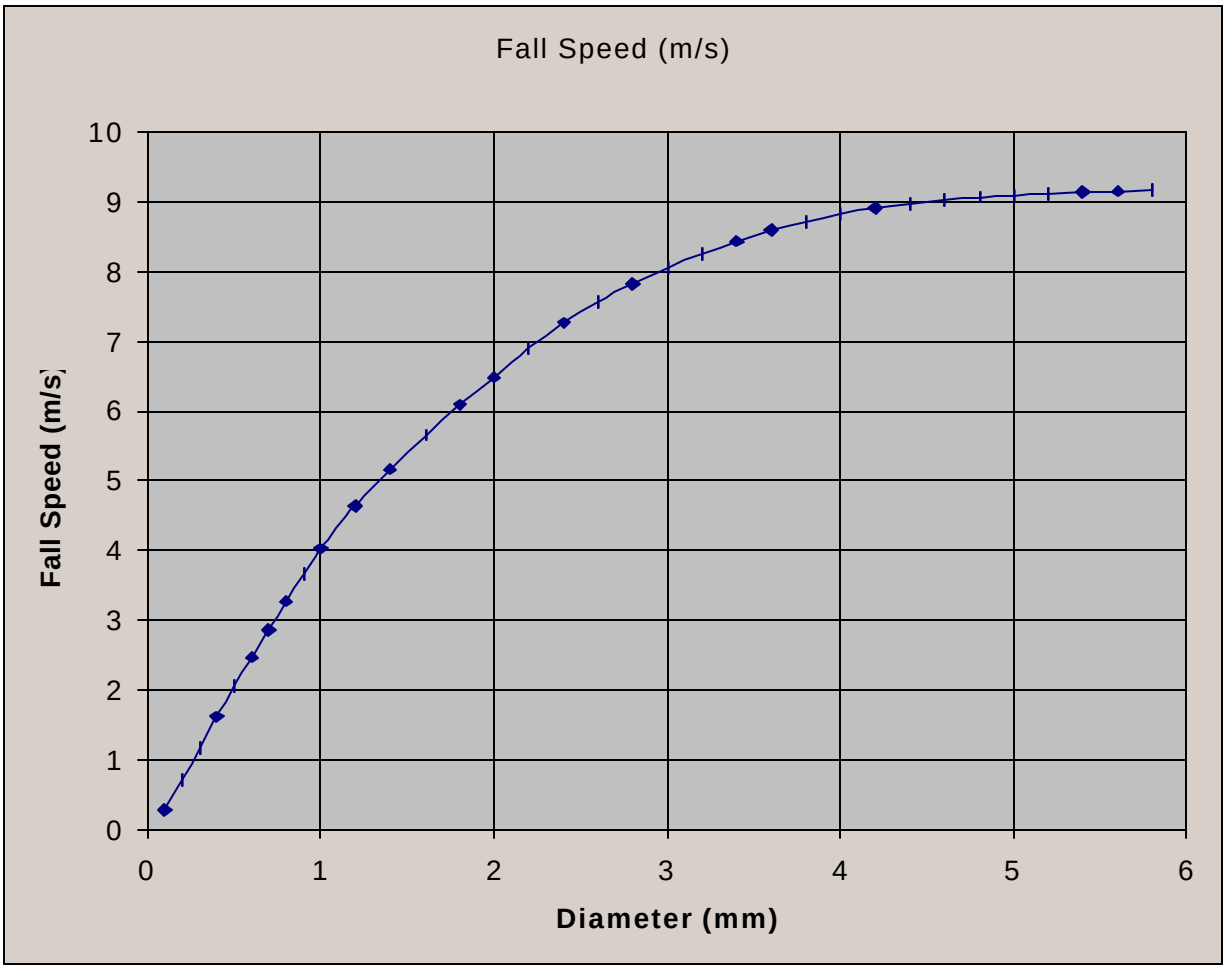
Atmospheric pressure is selectable, but constant for the whole model height

Saturation vapor pressure is obtained from an empirical formula

# Appendix 2: Terminal Velocity Data

TABLE 8.1. *Terminal Fall Speed as a Function of Drop Size (equivalent spherical diameter) (From Gunn and Kinzer, 1949)*

| Diam. (mm) | Fall speed (m/s) | Diam. (mm) | Fall speed (m/s) |
|---|---|---|---|
| 0.1 | 0.27 | 2.6 | 7.57 |
| 0.2 | 0.72 | 2.8 | 7.82 |
| 0.3 | 1.17 | 3.0 | 8.06 |
| 0.4 | 1.62 | 3.2 | 8.26 |
| 0.5 | 2.06 | 3.4 | 8.44 |
| 0.6 | 2.47 | 3.6 | 8.60 |
| 0.7 | 2.87 | 3.8 | 8.72 |
| 0.8 | 3.27 | 4.0 | 8.83 |
| 0.9 | 3.67 | 4.2 | 8.92 |
| 1.0 | 4.03 | 4.4 | 8.98 |
| 1.2 | 4.64 | 4.6 | 9.03 |
| 1.4 | 5.17 | 4.8 | 9.07 |
| 1.6 | 5.65 | 5.0 | 9.09 |
| 1.8 | 6.09 | 5.2 | 9.12 |
| 2.0 | 6.49 | 5.4 | 9.14 |
| 2.2 | 6.90 | 5.6 | 9.16 |
| 2.4 | 7.27 | 5.8 | 9.17 |



Since no formula was presented that adequately described terminal fall speed for drops with a radius greater than 2mm, I used linear approximations of this tabulated data split into four regimes:

| Charted Data | | | Curve Fit Data | | |
|---|---|---|---|---|---|
| R [mm] | U [ms] | slope | linear | % error | |
| 2 | 6.49 | 1.57 | 6.49 | 0.00 | from 2 to 3mm |
| 2.2 | 6.9 | | 6.804 | 1.39 | u=6.49+(R-2)*slope |
| 2.4 | 7.27 | | 7.118 | 2.09 | |
| 2.6 | 7.57 | | 7.432 | 1.82 | |
| 2.8 | 7.82 | | 7.746 | 0.95 | |
| 3 | 8.06 | | 8.06 | 0.00 | |
| 3.2 | 8.26 | 0.7125 | 8.26 | 0.00 | from 3 to 4mm |
| 3.4 | 8.44 | | 8.4025 | -0.44 | u=8.26+(R-3.2)*slope |
| 3.6 | 8.6 | | 8.545 | -0.64 | |
| 3.8 | 8.72 | | 8.6875 | -0.37 | |
| 4 | 8.83 | | 8.83 | 0.00 | |
| 4.2 | 8.92 | 0.2125 | 8.92 | 0.00 | from 4 to 5mm |
| 4.4 | 8.98 | | 8.9625 | -0.19 | u=8.92+(R-4.2)*slope |
| 4.6 | 9.03 | | 9.005 | -0.28 | |
| 4.8 | 9.07 | | 9.0475 | -0.25 | |
| 5 | 9.09 | | 9.09 | 0.00 | |
| 5.2 | 9.12 | 0.083333 | 9.12 | 0.00 | from 5 to 6mm |
| 5.4 | 9.14 | | 9.136666 | -0.04 | u=9.12+(R-5.2)*slope |
| 5.6 | 9.16 | | 9.153333 | -0.07 | |
| 5.8 | 9.17 | | 9.169999 | -0.00 | |

# Appendix 3 The MATLAB CODE

```
% John Dumas
% Cloud Drop Model
% drop2.m MATLAB file
% Written for MR3421 Cloud Physics
% 14 June 1999

% close figures, clear variables, clear screen
close all;
clear all;
clc;

% display user instruction screen
disp('This program is designed to model the growth and motion of a ');
disp('single cloud drop. ');
disp(' ');
disp('In this version, the temperature is a uniform and unchanging  ');
disp('10 degrees C.  Other assumptions and the equations may be found ');
disp('in the program code and the accompanying paper.');
disp(' ');
disp('You may either choose to enter variable values or use the default ');
disp('set.  After the model has run and the figure window appears, ');
disp('you will have the option of seeing on-screen data, writing a  ');
disp('MATLAB diary file or zooming in on the figure. ');
disp(' ');

% define constants
g=9.8;                      % [m/s] gravity
k1=1.19e+06;                % cm scale (terminal velocity const)
k2=2.01e+03;                % cm scale (tvel const)
k3=8e+03;                   % cm scale (assumed) (tvel const)
r=10e-06;                   % [m] 10 micron ambient droplet radius
rhol=1000;                  % [kg/m3] assums drop is pure water
TC=10;                      % [C] temperature uniform and unchanging
K=2.48e-02;                 % [m2/s] Coeff of Diffusion of WV in air @10C/100kPa
D=2.36e-05;                 % [J/msK] Coeff of Thermal Conductivity of air @10C/100kPa
Rv=461.5;                   % [J/kgK] gas const for water vapor
L=2477;                     % [J/g] @10C, latent heat of vaporization

% initialize variables as required
ImpactFlag=0;               % false
MakeZoom=1;                 % allows while loop to execute first time

% input variables by user prompt or default
VarIn=input('(1) Input Variables, or (2) Default Values? (1 or 2) ');

if VarIn == 1 % user selects variables
   disp(' ');
   LengthOfRun=input('Enter length [s] of model run (default 120): ');
   %initialize the arrays based on the length of the run
   zpos=zeros(1,LengthOfRun);
   Rmm=zeros(1,LengthOfRun);
   dropTV=zeros(1,LengthOfRun);
   E=zeros(1,LengthOfRun);
   DeltaRm=zeros(1,LengthOfRun);
   CGP=zeros(1,LengthOfRun);

   disp(' ');
   Rmm(1)=input('Enter the initial drop radius [mm](default 1): ');
   disp(' ');
   updraft=input('Enter the constant updraft velocity [m/s](def 0.5): ');
   disp(' ');
   M=input('Enter cloud liquid water content M [kg/m3] (def 0.001): ');
   disp(' ');
   S=input('Enter in cloud saturation ratio S (def 1.01): ');
   disp(' ');
   CloudTop=input('Enter cloud top height [m] (default 1000m): ');
   disp(' ');
   CloudBase=input('Enter cloud base height [m] (default 500m): ');
```

```
    disp(' ');
    zpos(1)=input('Enter drop initial height [m] (default 750m): ');
    disp(' ');
    AvgMB(1)=input('Enter avg atmospheric presure [mb] (default 850mb): ');
    disp(' ');
    RHHigh=input('Enter rel humidity above the cloud [%] (default 50): ');
    RHHigh=RHHigh/100;
    disp(' ');
    RHLow=input('Enter rel humidity below the cloud [%] (default 80): ');
    RHLow=RHLow/100;

else  % automatically assign the default values
    LengthOfRun=120;
    %initialize the arrrays based on the length of the run
    zpos=zeros(1,LengthOfRun);
    Rmm=zeros(1,LengthOfRun);
    dropTV=zeros(1,LengthOfRun);
    E=zeros(1,LengthOfRun);
    DeltaRm=zeros(1,LengthOfRun);
    CGP=zeros(1,LengthOfRun);

    Rmm(1)=1;              % see user input section for comments
    updraft=0.5;
    M=.001;
    S=1.01;
    CloudTop=1000;
    CloudBase=500;
    zpos(1)=750;
    AvgMB=850;
    RHHigh=.5;
    RHLow=.8;

end % input section

%calculate some constant parameters
TK=TC+273.15;                          % [K] temperature in Kelvin
EsT=6.112*exp((17.67*TC)/(TC+243.5)); % [mb] saturation vapor pressure
EsTPa=EsT*100;                         % [Pa] converts millibars to Pascals
DinCloud=D*100/(AvgMB/10);             % converts D from table 100kPa to input
KinCloud=K*100/(AvgMB/10);             % converts K from table 100kPa to input
Fd=((rhol*Rv*TK)/(DinCloud*EsTPa))/(1000^2);                    % [s/mm2]
Fk= (((L*1000)/(Rv*TK))-1)*(L*1000*rhol/KinCloud*TK)/(1000^2); % [s/mm2]
NCGP=1/(Fk+Fd);                        % norm condensation growth param

% set up the plot
axis([0,LengthOfRun,0,CloudTop+500]);    % 0-LOR sec, 0-Top+500m ht
xlabel('Time [s]');
ylabel('Vertical Position Z [m]');
title(['Drop motion for initial radius= ',num2str(Rmm(1)),' mm']);

% Draw in the cloud layer
for time=1:LengthOfRun
  text(time,CloudTop,'-','Color',['Blue']);
  text(time,CloudBase,'-','Color',['Blue']);
end % end drawing cloud line loop
text(1,CloudTop+20,'Cloud Top','Color',['Blue']);
text(1,CloudBase-20,'Cloud Base','Color',['Blue']);

for time=1:LengthOfRun
   % at the start if each second, begin by assigning the Rmm the ENDING
   % value from the previous second. (except at time=1)
   if time ==1
      Rmm(time)=Rmm(1);
   else
      Rmm(time)=Rmm(time-1);
   end % end Rmm bookkeeping if block

   % loop begins knowing radius(mm) but some formulas need other units
   Rm(time)=Rmm(time)/1000;                    % converts to m
   Rcm(time)=Rm(time)*100;                     % converts to cm
   Rmic(time)=Rmm(time)*1000;                  % converts to microns
```

```
% Calculate the terminal velocity of the drop based on its radius
% from the appropriate formula.  Assume that the drop INSTANTLY
% achieves this velocity.

if Rmm(time)< 0.02                          % if r< 20 microns
   dropTV(time)=(k1*Rcm(time)^2)/100;       % calc & convert cm to m/s
elseif Rmm(time)>= 0.02 & Rmm(time) <= 0.05  % 20-50 micron transition
   TV1=(k1*Rcm(time)^2)/100;
   TV2=(k2*sqrt(Rcm(time)))/100;
   dropTV(time)=TV1+((Rmic(time)-20)/30)*(TV2-TV1);
elseif Rmm(time) >= 0.05 & Rmm(time) <= 0.6  % 50 mic<r<0.6mm
   dropTV(time)=(k2*sqrt(Rcm(time)))/100;
elseif Rmm(time) > 0.6 & Rmm(time) <2        % 0.6mm<r<2mm
   dropTV(time)=(k3*Rcm(time))/100;
elseif Rmm(time) >= 2 & Rmm(time) <=3
   dropTV(time)=6.49+(Rmm(time)-2)*1.57;     % >2mm, linear rels
elseif Rmm(time) > 3 & Rmm(time) <=4
   dropTV(time)=8.26+(Rmm(time)-3.2)*0.7125;
elseif Rmm(time) > 4 & Rmm(time) <=5
   dropTV(time)=8.92+(Rmm(time)-4.2)*0.2125;
else                                        % Rmm>5mm
   dropTV(time)=9.12+(Rmm(time)-5.2)*0.0833;
end % end calculation TV block

% Plot the Z position of this drop, based on terminal velocity down
% and updraft up.  The size of the drop is noted through font size.
if time==1
   zpos(time)=zpos(1)-dropTV(time)+updraft;
else
   if zpos(time-1)<=0                      % if ever 0, stay 0
      zpos(time)=0;
   else
      zpos(time)=zpos(time-1)-dropTV(time)+updraft;
   end
end % end z-position block

% the lowest the drop can go is the ground, make "impact" info
if zpos(time)<= 0
   zpos(time)=0;            % no lower than the ground
   ImpactFlag=ImpactFlag+1; % 1st time makes true
   if ImpactFlag==1         % should only execute once
      ITime=time;
      IRadius=Rmm(time);
      text(time-20,550,['Impact at ',num2str(ITime),' seconds'],...
         'Color',['Red']);
      text(time-20,450,['Radius = ',num2str(IRadius),' mm'],...
         'Color',['Red']);
   end % end one time impact section
end % end zpos zero block

% Collision Process only will occur if the drop is within the cloud
if zpos(time) >= CloudBase & zpos(time) <= CloudTop
   % Determine what growth will occur due to collisions.  This is done
   % by assuming all other cloud droplets are uniformly 10 microns and
   % looking up the collision Efficiency E from R&Y table 8.2, then
   % applying R&Y equation 8.15
   if Rmic(time) <=10                           % if r < 10 microns
      E(time)=0;
   elseif Rmic(time) > 10 & Rmic(time) <= 20
      E(time)=0.17;
   elseif Rmic(time) > 20 & Rmic(time) <= 30
      E(time)=0.37;
   elseif Rmic(time) > 30 & Rmic(time) <= 40
      E(time)=0.55;
   elseif Rmic(time) > 40 & Rmic(time) <= 50
      E(time)=0.58;
   elseif Rmic(time) > 50 & Rmic(time) <= 60
      E(time)=0.68;
   elseif Rmic(time) > 60 & Rmic(time) <= 80
      E(time)=0.76;
```

```
      elseif Rmic(time) > 80 & Rmic(time) <= 100
         E(time)=0.81;
      elseif Rmic(time) > 100 & Rmic(time) <= 150
         E(time)=0.83;
      elseif Rmic(time) > 150 & Rmic(time) <= 200
         E(time)=0.87;
      elseif Rmic(time) > 200 & Rmic(time) <= 300
         E(time)=0.87;
      elseif Rmic(time) > 300 & Rmic(time) <= 400
         E(time)=0.88;
      elseif Rmic(time) > 400 & Rmic(time) <= 500
         E(time)=0.88;
      elseif Rmic(time) > 500 & Rmic(time) <= 600
         E(time)=0.88;
      elseif Rmic(time) > 600 & Rmic(time) <= 1000
         E(time)=0.88;
      elseif Rmic(time) > 1000 & Rmic(time) <= 1400
         E(time)=0.88;
      elseif Rmic(time) > 1400 & Rmic(time) <= 1800
         E(time)=0.86;
      elseif Rmic(time) > 1800 & Rmic(time) <= 2400
         E(time)=0.83;
% the table only went to 3000 microns, but I want to go to 6000
% so I cheated and made all values >2400 the same. ASSUMPTION
      elseif Rmic(time) > 2400
         E(time)=0.81;
      end % E table ifs

      DeltaRm(time)=1*(E(time)*M*dropTV(time))/(4*rhol);
      % since dropTV is in m/s, and the units on M and rhol cancel, dR/dT
      % is in m/s.  Since the time step in this model is 1 sec, the result
      % is dR in meters.

      % compute the new radius based on old radius and growth by collision
      if time ==1
         Rmm(time)=Rmm(1)+(DeltaRm(time)*1000);
      else
         Rmm(time)=Rmm(time-1)+(DeltaRm(time)*1000);
         if zpos(time)==0
            Rmm(time)=0;
         end % end zpos=0 block
      end % end change in Rmm for collision block
   end % in-cloud collision block

   % determine the radius change due to condensation/evaporation
   if zpos(time)>CloudTop
      SAT=RHHigh;                  % saturation above cloud
   elseif zpos(time)<CloudBase
      SAT=RHLow;                   % saturation below cloud
   else
      SAT=S;                       % variable saturation within the cloud
   end % end SAT selection block

   % condensation growth/evaporation section (7.19 from R&Y)
   if zpos(time)==0
      cgp(time)=0;
   else
      CGP(time)=(SAT-1)*NCGP;
   end
   arg=(Rmm(time)^2+(2*CGP(time)*1));

   % finally, the Rmm at the end of the loop is...
   if arg<=0                       % avoiding complex roots
      Rmm(time)=0;                 % zero if it evaporated away
   else
      Rmm(time)=sqrt(arg);         % or sum collisions & condensation arg
      if zpos(time)==0
            Rmm(time)=0;           % unless it hit ground
      end % end zpos=0 block
   end % end growth/evaporation by condensation block
```

```
end %end of LengthOfRun loop

% draw the drop at height and time position
LargestDrop=max(Rmm);
SmallestDrop=min(Rmm);
DeltaDrop=LargestDrop-SmallestDrop;
for time=1:LengthOfRun
   if Rmm(time)==0
      FSize=1;
   else
      FSize=2+((Rmm(time)-SmallestDrop)/DeltaDrop)*8;
   end % end font scale block
   text(time,zpos(time),'o','Fontsize',[FSize]);
end % drawing loop

% bookkeeping section

% zoom-in on the figure window?
disp(' ');
while MakeZoom == 1
   MakeZoom=input('Do you want to zoom to a time interval? 1=Yes, 2=No: ');
   if MakeZoom == 1
      disp(' ');
      disp('WARNING: choosing values outside length of run = CRASH');
      disp('Looking at Figure 1...');
      STime=input('Enter the starting time [s] (MUST BE>1): ');
      ETime=input('Enter the ending time of interest [s]: ');
      figure
      ZoomZpos=zpos(STime:ETime);
      zbottom=min(ZoomZpos);
      ztop=max(ZoomZpos);
      axis([STime,ETime,zbottom,ztop]);
      xlabel('Time [s]');
      ylabel('Vertical Position Z [m]');
      title('Zoom Plot');
      for ztime=STime:ETime
         if CloudTop<ztop & CloudTop>zbottom
            text(ztime,CloudTop,'-','Color',['Blue']);
         end
         if CloudBase<ztop & CloudBase>zbottom
            text(ztime,CloudBase,'-','Color',['Blue']);
         end
         if Rmm(ztime)==0
            FSize=1;
         else
            FSize=2+((Rmm(ztime)-SmallestDrop)/DeltaDrop)*8;
         end % end font scale block
         text(ztime,zpos(ztime),'o','Fontsize',[FSize]);
      end
      text(STime,CloudTop+20,'Cloud Top','Color',['Blue']);
      text(STime,CloudBase-20,'Cloud Base','Color',['Blue']);
      if ImpactFlag~=0 & STime<ITime & ETime>ITime
         text(ITime,550,['Impact at ',num2str(ITime),' seconds'],...
            'Color',['Red']);
         text(ITime,450,['Radius = ',num2str(IRadius),' mm'],...
            'Color',['Red']);
      end %impact plot
   end %this iteration of zoomplot
end %the zoom routine

% on-screen listing?
clc;
disp('The following information applies for this run:');
fprintf('The constant updraft velocity [m/s]      %11.2f\n',updraft);
fprintf('The cloud liquid water content M [kg/m3] %12.3f\n',M);
fprintf('The cloud top height [m]                 %10.1f\n',CloudTop);
fprintf('The cloud base height [m]                %10.1f\n',CloudBase);
fprintf('The avg atmospheric pressure [mb]         %10.1f\n',AvgMB);
fprintf('The cloud saturation ratio S             %11.2f\n',S);
fprintf('The rel above cloud humidity [percent]   %11.2f\n',RHHigh);
fprintf('The rel below cloud humidity [percent]   %11.2f\n\n',RHLow);
```

```
MakeList=input('Do you want an on-screen data listing? 1=Yes, 2=No: ');
if MakeList == 1
   disp('Time    Z Pos        Rmm    Col Eff(E) CondGrowth     TermVel')
   for time=1:LengthOfRun
      fprintf('%3d %10.3f %10.3f %10.3f   %10.3e %10.3f\n', ...
         time,zpos(time),Rmm(time),E(time),CGP(time),dropTV(time));
   end
end

% diary file?
disp(' ');
MakeDiary=input('Make a diary file of the list? 1=Yes, 2=No: ');
if MakeDiary == 1
   disp('The Diary file will be placed in the default directory.');
   DName=input('Name the diary file (without path info): ','s');
   diary (DName)
   disp('The following information applies for this run:');
   fprintf('The constant updraft velocity [m/s]      %11.2f\n',updraft);
   fprintf('The cloud liquid water content M [kg/m3] %12.3f\n',M);
   fprintf('The cloud top height [m]                 %10.1f\n',CloudTop);
   fprintf('The cloud base height [m]                %10.1f\n',CloudBase);
   fprintf('The avg atmospheric pressure [mb]         %10.1f\n',AvgMB);
   fprintf('The cloud saturation ratio S             %11.2f\n',S);
   fprintf('The rel above cloud humidity [percent]   %11.2f\n',RHHigh);
   fprintf('The rel below cloud humidity [percent]   %11.2f\n\n',RHLow);
   disp('Time    Z Pos        Rmm    Col Eff(E) CondGrowth     TermVel')
   for time=1:LengthOfRun
      fprintf('%3d %10.3f %10.3f %10.3f   %10.3e %10.3f\n', ...
         time,zpos(time),Rmm(time),E(time),CGP(time),dropTV(time));
   end
   diary off
end
```